<u>Attachment B</u>

Pages 49, 50, 54, and 103-109 of the Microfiche Appendix included in U.S. Patent App. Ser. No. 08/516,036

(11 pages)

IS 1166194

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

May 26, 2004

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

The Appendix Microfiche
SERIAL NUMBER: 08/516,036
FILING DATE: August 16, 1995

By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

[illegible]
M. K. HAWKINS
Certifying Officer

Instruction Set

All instructions are 32 bits in size, and use the high order 8 bits to specify a major operation code.

31 24	23 0
major	other
8	24

The major field is filled with a value specified by the following table:[5]

MAJOR	0	32	64	96	128	160	192	224
0	ERES	GSHUFFLEI	FMULADD16	GMULADD1	LU16LAI	SAAS64LAI	EADDIO	BFE16
1	ESHUFFLE4MUX	GSHUFFLE4MUX	FMULADD32	GMULADD2	LU16BAI	SAAS64BAI	EADDIUO	BFNUE16
2		GSELECT8	FMULADD64	GMULADD4	LU16LI	SCAS64LAI	ESETIL	BFNUGE16
3	EMDEPI	GMDEPI		GMULADD8	LU16BI	SCAS64BAI	ESETIGE	BFNUL16
4	EMUX	GMUX	FMULSUB16	GMULADD16	LU32LAI	SMAS64LAI	ESETIE	BFE32
5	E8MUX	G8MUX	FMULSUB32	GMULADD32	LU32BAI	SMAS64BAI	ESETINE	BFNUE32
6	EGFMUL64	GGFMUL8	FMULSUB64	GMULADD64	LU32LI	SMUX64LAI	ESETIUL	BFNUGE32
7	ETRANSPOSE8MUX	GTRANSPOSE8MUX		GEXTRACT128	LU32BI	SMUX64BAI	ESETIUGE	BFNUL32
8					L16LAI	S16LAI	ESUBIO	BFE64
9	ESWIZZLE	GSWIZZLE		GUMULADD2	L16BAI	S16BAI	ESUBIUO	BFNUE64
10		GSWIZZLECOPY		GUMULADD4	L16LI	S16LI	ESUBIL	BFNUGE64
11		GSWIZZLESWAP		GUMULADD8	L16BI	S16BI	ESUBIGE	BFNUL64
12	EDEPI	GDEPI	F.16	GUMULADD16	L32LAI	S32LAI	ESUBIE	BFE128
13	EUDEPI	GUDEPI	F.32	GUMULADD32	L32BAI	S32BAI	ESUBINE	BFNUE128
14	EWTHI	GWTHI	F.64	GUMULADD64	L32LI	S32LI	ESUBIUL	BFNUGE128
15	EUWTHI	GUWTHI		GUEXTRACTI128	L32BI	S32BI	ESUBIUGE	BFNUL128
16			GFMULADD16	GEXTRACTI	L64LAI	S64LAI	EADDI	BANDE
17			GFMULADD32	GEXTRACTI16	L64BAI	S64BAI	EXORI	BANDNE
18			GFMULADD64	GEXTRACTI32	L64LI	S64LI	EORI	BL/BLZ
19			GFMULADD128	GUEXTRACTI64	L64BI	S64BI	EANDI	BGE/BGEZ
20			GFMULSUB16	GEXTRACT	L128LAI	S128LAI	ESUBI	BE
21			GFMULSUB32	.I.64	L128BAI	S128BAI		BNE
22			GFMULSUB64	G EXTRACT	L128LI	S128LI	ENORI	BUL/BGZ
23			GFMULSUB128	I 128	L128BI	S128BI	ENANDI	BUGE/BLEZ
24				G 1	L8I	S8I		BGATEI
25				G 2	LU8I			
26				G 4				
27				G 8				
28		ECOPYI	GF.16	G 16			ECOPYI	BI
29			GF.32	G 32				BLINKI
30			GF.64	G 64				
31		E.MINOR	GF.128	G.128	L.MINOR	S.MINOR	E.MINOR	B.MINOR

major operation code field values

For the major operation field values A.MINOR, L.MINOR, E.MINOR, F.16, F.32, F.64, F.128, GF.16, GF.32, GF.64, G.1, G.2, G.4, G.8, G.16, G.32, G.64, S.MINOR and B.MINOR, the lowest-order six bits in the instruction specify a minor operation code:

31 24	23 6	5 0
major	other	minor
8	18	6

[5]Blank table entries cause the Reserved Instruction exception to occur.

The minor field is filled with a value from one of the following tables:

E.MINOR	0	8	16	24	32	40	48	56
0	EADDO	ESUBO	EANDN		EADD	ESUB	ESHLIO	ESHRI
1	EADDUO	ESUBUO	EXOR		ESHLO	ESHLUO		
2	ESETL	ESUBL	EOR				ESHLIUO	EUSHRI
3	ESETGE	ESUBGE	EAND		ELMS	EULMS		
4	ESETE	ESUBE	EORN		EASUM	ESELECT8	ESHUFFLEI	ERORTI
5	ESETNE	ESUBNE	EXNOR		EROTL	ESHL		
6	ESETUL	ESUBUL	ENOR		ESHR	EUSHR	ESHLI	EMSHRI
7	ESETUGE	ESUBUGE	ENAND		EROTR	EMSHR		

minor operation code field values for E.MINOR

F.size	0	8	16	24	32	40	48	56
0	FADD.N	FADD.T	FADD.F	FADD.C	FADD	FADD.X	FSETE	FSETE X
1	FSUB N	FSUB.T	FSUB F	FSUB C	FSUB	FSUB.X	FSETNUE	FSETNUE X
2	FMUL N	FMUL T	FMUL F	FMUL.C	FMUL	FMUL.X	FSETNUGE	FSETNGE X
3	FDIV.N	FDIV.T	FDIV.F	FDIV C	FDIV	FDIV X	FSETNUL	FSETNUL X
4	F UNARY.N	F.UNARY T	F.UNARY.F	F.UNARY C	F UNARY	F UNARY.X		
5								
6								
7								

minor operation code field values for F.size

GF.size	0	8	16	24	32	40	48	56
0	GFADD.N	GFADD.T	GFADD.F	GFADD.C	GFADD	GFADD.X	GFSETE	GFSETE X
1	GFSUB N	GFSUB T	GFSUB.F	GFSUB C	GFSUB	GFSUB X	GFSETNUE	GFSETNUE X
2	GFMUL.N	GFMUL.T	GFMUL F	GFMUL.C	GFMUL	GFMUL.X	GFSETNUGE	GFSETNGE.X
3	GFDIV N	GFDIV.T	GFDIV F	GFDIV C	GFDIV	GFDIV.X	GFSETNUL	GFSETNUL X
4	GF UNARY.N	GF.UNARY.T	GF.UNARY F	GF.UNARY C	GF.UNARY	GF.UNARY X		
5								
6								
7								

minor operation code field values for GF.size

G size	0	8	16	24	32	40	48	56
0		GMUL	GANDN		GADD	GSUB	GEXPAND	GSHR
1		GUMUL	GXOR		GCOMPRESS	GUCOMPRESS	I	I
2	GSETL	GDIV	GOR				GUEXAPAND	GUSHR
3	GSETGE	GUDIV	GAND				I	I
4	GSETE	GSUB	GORN		GEXPAND	GUEXPAND	GCOMPRESSI	GROTR
5	GSETNE		GXNOR		GROTL	GSHL	GUCOMPRESSI	.I
6	GSETUL		GNOR		GSHR	GUSHR	GSHI	GMSHR
7	GSETUGE		GNAND		GROTR	GMSHR	.I	I

minor operation code field values for G.size

L.MINOR	0	8	16	24	32	40	48	56
0	LU16LA	L16LA	L64LA	L8				
1	LU16BA	L16BA	L64BA	LU8				
2	LU16L	L16L	L64L					
3	LU16B	L16B	L64B					
4	LU32LA	L32LA	L128LA					
5	LU32BA	L32BA	L128BA					
6	LU32L	L32L	L128L					
7	LU32B	L32B	L128B					

minor operation code field values for L.MINOR

```
GUMULADD2, GUMULADD4,
GUMULADD8, GUMULADD16, GUMULADD32,
GMUX, GMUXGATHER, GSCATTERMUX, G.EXTRACT.128:
    GroupTernary(major,size,ra,rb,rc,rd)
G.EXTRACT.I, G.EXTRACT.I.64:
    GroupExtractImmediate(major,ra,rb,rc,minor)
G.1, G.2, G.4, G.8, G.16, G.32:
    case minor of
        G.SHL, G.SHR, G.USHR, G.ADD, G.SUB, G.MUL, G.UMUL,
        G.AND, G.OR, G.XOR, G.ANDN, G.NAND, G.NOR, G.XNOR, G.ORN,
        G.SET.E, G.SET.NE, G.SET.L, G.SET.GE, G.SET.UL, G.SET.UGE,
        G.COPY, G.SWAP, G.DEAL, G.SHUFFLE, G.COMPRESS, G.EXPAND,
        G.GATHER, G.SCATTER:
            Group(minor,major,ra,rb,rc)
        G.COMPRESS.I, G.EXPAND.I, G.SHL.I, G.SHR.I, G.U.SHR.I:
            GroupShortImmediate(minor,major,ra,simm,rc)
        G.EXTRACT.I:
            GroupExtractImmediate(major,ra,rb,rc,minor)
        others:
            raise ReservedInstruction
    endcase
GFMULADD16, GFMULADD32, GFMULADD64,
GFMULSUB16, GFMULSUB32, GFMULSUB64:
    GroupFloatingPointTernary(major,ra,rb,rc,rd)
GF.16, GF.32, GF.64, GF.128:
    case minor of
        GF.ADD.N, GF.SUB.N, GF.MUL.N, GF.DIV.N,
        GF.ADD.T, GF.SUB.T, GF.MUL.T, GF.DIV.T,
        GF.ADD.F, GF.SUB.F, GF.MUL.F, GF.DIV.F,
        GF.ADD.C, GF.SUB.C, GF.MUL.C, GF.DIV.C,
        GF.ADD, GF.SUB, GF.MUL, GF.DIV,
        GF.ADD.X, GF.SUB.X, GF.MUL.X, GF.DIV.X,
        GF.SET.E, GF.SET.NE, GF.SET.UE, GF.SET.NUE,
        GF.SET.NUGE, GF.SET.UGE, GF.SET.UL, GF.SET.NUL,
        GF.SET.E.X, GF.SET.NE.X, GF.SET.UE.X, GF.SET.NUE.X,
        GF.SET.L.X, GF.SET.NL.X, GF.SET.NGE.X, GF.SET.GE.X:
            GroupFloatingPoint(minor.op, major.size, minor.round, ra, rb, rc)
        GF.UNARY.N, GF.UNARY.T, GF.UNARY.F, GF.UNARY.C,
        GF.UNARY, GF.UNARY.X:
            case unary of
                GF.ABS, GF.NEG, GF.SQR,
                GF.HALF, GF.SINGLE, GF.DOUBLE, GF.QUAD,
                GF.INT, GF.FLOAT:
                    GroupFloatingPointUnary(unary.op, major.size,
                            minor.round, ra, rc)
                others:
                    raise ReservedInstruction
            endcase
        others:
            raise ReservedInstruction
    endcase
L.MINOR
    case minor of
        L16L, LU16L, L32L, LU32L, L64L, L128L, L8, LU8,
        L16LA, LU16LA, L32LA, LU32LA, L64LA, L128LA,
        L16B, LU16B, L32B, LU32B, L64B, L128B,
        L16BA, LU16BA, L32BA, LU32BA, L64BA, L128BA:
            Load(minor,ra,rb,rc)
```

II 14

Group

These instructions take two operands, perform a group of operations on partitions of bits in the operands, and catenate the results together .

Operation codes

G.ADD.2	Group add pecks
G.ADD.4	Group add nibbles
G.ADD.8	Group add bytes
G.ADD.16	Group add doublets
G.ADD.32	Group add quadlets
G.ADD.64	Group add octlets
G.AND[14]	Group and
G.ANDN[15]	Group and not
G.COMPRESS.1	Group compress bits
G.COMPRESS.2	Group compress pecks
G.COMPRESS.4	Group compress nibbles
G.COMPRESS.8	Group compress bytes
G.COMPRESS.16	Group compress doublets
G.COMPRESS.32	Group compress quadlets
G.COMPRESS.64	Group compress octlets
G.DIV.64	Group signed divide octlets
G.EXPAND.1	Group signed expand bits
G.EXPAND.2	Group signed expand pecks
G.EXPAND.4	Group signed expand nibbles
G.EXPAND.8	Group signed expand bytes
G.EXPAND.16	Group signed expand doublets
G.EXPAND.32	Group signed expand quadlets
G.EXPAND.64	Group signed expand octlet
G.GATHER.2	Group gather pecks
G.GATHER.4	Group gather nibbles
G.GATHER.8	Group gather bytes
G.GATHER.16	Group gather doublets
G.GATHER.32	Group gather quadlets
G.GATHER.64	Group gather octlets
G.GATHER.128[16]	Group gather hexlets
G.MUL.1[17]	Group signed multiply bits
G.MUL.2	Group signed multiply pecks
G.MUL.4	Group signed multiply nibbles
G.MUL.8	Group signed multiply bytes

[14]G.AND does not require a size specification, and is encoded as G.AND.1.
[15]G.ANDN does not require a size specification, and is encoded as G.ANDN.1. G.ANDN is used as the encoding for G.SET.L.1, and by reversing the operands, for G.SET.UL.1.
[16]G.GATHER.128 is encoded as G.GATHER.1
[17]G.MUL.1 is used as the encoding for G.UMUL.1.

G.MUL.16	Group signed multiply doublets
G.MUL.32	Group signed multiply quadlets
G.MUL.64	Group signed multiply octlets
G.NAND[18]	Group nand
G.NOR[19]	Group nor
G.OR[20]	Group or
G.ORN[21]	Group or not
G.POLY.1	Group polynomial divide bits
G.POLY.2	Group polynomial divide pecks
G.POLY.4	Group polynomial divide nibbles
G.POLY.8	Group polynomial divide bytes
G.POLY.16	Group polynomial divide doublets
G.POLY.32	Group polynomial divide quadlets
G.POLY.64	Group polynomial divide octlets
G.ROTL.2	Group rotate left pecks
G.ROTL.4	Group rotate left nibbles
G.ROTL.8	Group rotate left bytes
G.ROTL.16	Group rotate left doublets
G.ROTL.32	Group rotate left quadlets
G.ROTL.64	Group rotate left octlets
G.ROTL.128	Group rotate left hexlets
G.ROTR.2	Group rotate right pecks
G.ROTR.4	Group rotate right nibbles
G.ROTR.8	Group rotate right bytes
G.ROTR.16	Group rotate right doublets
G.ROTR.32	Group rotate right quadlets
G.ROTR.64	Group rotate right octlets
G.ROTR.128	Group rotate right hexlets
G.SCATTER.2	Group scatter pecks
G.SCATTER.4	Group scatter nibbles
G.SCATTER.8	Group scatter bytes
G.SCATTER.16	Group scatter doublets
G.SCATTER.32	Group scatter quadlets
G.SCATTER.64	Group scatter octlets
G.SCATTER.128[22]	Group scatter hexlet
G.SHL.2	Group shift left pecks
G.SHL.4	Group shift left nibbles
G.SHL.8	Group shift left bytes
G.SHL.16	Group shift left doublets
G.SHL.32	Group shift left quadlets

[18]G.NAND does not require a size specification, and is encoded as G.NAND.1.
[19]G.NOR does not require a size specification, and is encoded as G.NOR.1.
[20]G.OR does not require a size specification, and is encoded as G.OR.1.
[21]G.ORN does not require a size specification, and is encoded as G.ORN.1. G.ORN is used as the encoding for G.SET.UGE.1, and by reversing the operands, for G.SET.GE.1.
[22]G.SCATTER.128 is encoded as G.SCATTER.1

G.SHL.64	Group shift left octlets
G.SHL.128	Group shift left hexlets
G.SHR.2	Group signed shift right pecks
G.SHR.4	Group signed shift right nibbles
G.SHR.8	Group signed shift right bytes
G.SHR.16	Group signed shift right doublets
G.SHR.32	Group signed shift right quadlets
G.SHR.64	Group signed shift right octlets
G.SHR.128	Group signed shift right hexlets
G.U.DIV.64	Group signed divide octlets
G.U.EXPAND.1	Group unsigned expand bits
G.U.EXPAND.2	Group unsigned expand pecks
G.U.EXPAND.4	Group unsigned expand nibbles
G.U.EXPAND.8	Group unsigned expand bytes
G.U.EXPAND.16	Group unsigned expand doublets
G.U.EXPAND.32	Group unsigned expand quadlets
G.U.EXPAND.64	Group unsigned expand octlet
G.U.MUL.2	Group unsigned multiply pecks
G.U.MUL.4	Group unsigned multiply nibbles
G.U.MUL.8	Group unsigned multiply bytes
G.U.MUL.16	Group unsigned multiply doublets
G.U.MUL.32	Group unsigned multiply quadlets
G.U.MUL.64	Group unsigned multiply octlets
G.U.SHR.2	Group unsigned shift right pecks
G.U.SHR.4	Group unsigned shift right nibbles
G.U.SHR.8	Group unsigned shift right bytes
G.U.SHR.16	Group unsigned shift right doublets
G.U.SHR.32	Group unsigned shift right quadlets
G.U.SHR.64	Group unsigned shift right octlets
G.U.SHR.128	Group unsigned shift right hexlets
G.XNOR[23]	Group exclusive-nor
G.XOR[24]	Group exclusive-or

[23]G.XNOR does not require a size specification, and is encoded as G.XNOR.1. G.XNOR is used as the encoding for G.SET.E.1.
[24]G.XOR does not require a size specification, and is encoded as G.XOR.1. G.XOR is used as the encoding for G.ADD.1, G.SUB.1 and G.SET.NE.1.

class	op	size
linear	ADD	2 4 8 16 32 64
bitwise	AND ANDN NAND NOR OR ORN XNOR XOR	
signed multiply	MUL	1 2 4 8 16 32 64
unsigned multiply	U.MUL	2 4 8 16 32 64
signed divide	DIV	64
unsigned divide	U.DIV	64
	GATHER SCATTER	2 4 8 16 32 64
galois field	POLY	1 2 4 8 16 32 64
precision	COMPRESS EXPAND U.EXPAND	1 2 4 8 16 32 64
shift	ROTR ROTL SHR SHL U.SHR	2 4 8 16 32 64 128

Format

G.op.size rc=ra,rb

31 24	23 18	17 12	11 6	5 0
G.size	**ra**	**rb**	**rc**	**op**
8	6	6	6	6

Description

Two values are taken from the contents of registers or register pairs specified by ra and rb. The specified operation is performed, and the result is placed in the register or register pair specified by rc.

A reserved instruction exception occurs if rc_0 is set, and for certain operations, if ra_0 or rb_0 is set.

Definition

```
def Group(op,size,ra,rb,rc)
    case op of
        G.MUL, G.U.MUL, G.DIV, G.U.DIV:
            a ← RegRead(ra, 64)
            b ← RegRead(rb, 64)
        G.ADD, G.SUB, G.SET.L, G.SET.UL, G.SET.E, G.SET.NE, G.SET.GE, G.SET.UGE,
        G.AND, G.OR, G.XOR, G.ANDN, G.NAND, G.NOR, G.XNOR, G.ORN,
        G.GATHER, G.SCATTER:
            a ← RegRead(ra, 128)
            b ← RegRead(rb, 128)
        G.COMPRESS, G.ROTL, G.ROTR, G.SHL, G.SHR, G.U.SHR, G.POLY:
            a ← RegRead(ra, 128)
            b ← RegRead(rb, 64)
        G.EXPAND, G.U.EXPAND:
            a ← RegRead(ra, 64)
            b ← RegRead(rb, 64)
    endcase
    case op of
        G.ADD:
            for i ← 0 to 128-size by size
                $c_{i+size-1..i} \leftarrow a_{i+size-1..i} + b_{i+size-1..i}$
            endfor
        G.MUL:
            for i ← 0 to 64-size by size
                $c_{2*(i+size)-1..2*i} \leftarrow (a_{size-1}^{size} \| a_{size-1+i..i}) * (b_{size-1}^{size} \| b_{size-1+i..i})$
            endfor
        G.U.MUL:
            for i ← 0 to 64-size by size
                $c_{2*(i+size)-1..2*i} \leftarrow (0^{size} \| a_{size-1+i..i}) * (0^{size} \| b_{size-1+i..i})$
            endfor
        G.DIV:
            if (b = 0) or ( (a = (1||0^{63})) and (b = 1^{64}) ) then
                c ← undefined
            else
                q ← a / b
                r ← a - q*b
                $c \leftarrow r_{63..0} \| q_{63..0}$
            endif
        G.U.DIV:
            if b = 0 then
                c ← undefined
            else
                q ← (0 || a) / (0 || b)
                r ← a - q*b
                $c \leftarrow r_{63..0} \| q_{63..0}$
            endif
        G.AND:
            c ← a and b
        G.OR:
            c ← a or b
        G.XOR:
            c ← a xor b:
        G.ANDN:
            c ← a and not b
        G.NAND:
            c ← not (a and b)
        G.NOR:
            c ← not (a or b)
        G.XNOR:
            c ← not (a xor b)
        G.ORN:
            c ← a or not b
        G.POLY:
            p[0] ← a
            for i ← 1 to size
                $p[i] \leftarrow (p[i-1]_0\ ?\ (0^{64} \| b) : 0^{128})$ xor $(p[i-1]_0 \| p[i-1]_{127..1})$
            endfor
            c ← p[size]
        G.GATHER:
            for k ← 0 to 128-size by size
                j ← k
                for i ← k to k+size-1 by 1
```

A 12

```
            if $a_i$ then
                $c_j \leftarrow b_i$
                $j \leftarrow j + 1$
            endif
        endfor
        $j \leftarrow k+size-1$
        for $i \leftarrow k+size-1$ to k by -1
            if $\sim a_i$ then
                $c_j \leftarrow b_i$
                $j \leftarrow j - 1$
            endif
        endfor
    endfor
G.SCATTER:
    for $k \leftarrow 0$ to 128-size by size
        $j \leftarrow k$
        for $i \leftarrow k$ to k+size-1 by 1
            if $a_i$ then
                $c_i \leftarrow b_j$
                $j \leftarrow j + 1$
            endif
        endfor
        $j \leftarrow k+size-1$
        for $i \leftarrow k+size-1$ to k by -1
            if $\sim a_i$ then
                $c_i \leftarrow b_j$
                $j \leftarrow j - 1$
            endif
        endfor
    endfor
G.COMPRESS:
    for $i \leftarrow 0$ to 64-size by size
        $c_{i+size-1..i} \leftarrow a_{i+i+size-1+(b\&(size-1))..i+i+(b\&(size-1))}$
    endfor
G.EXPAND:
    for $i \leftarrow 0$ to 64-size by size
        $c_{i+i+size+size-1..i+i} \leftarrow a_{i+size-1}^{size-(b\&(size-1))} \| a_{i+size-1..i} \| 0^{b\&(size-1)}$
    endfor
G.U.EXPAND:
    for $i \leftarrow 0$ to 64-size by size
        $c_{i+i+size+size-1..i+i} \leftarrow 0^{size-(b\&(size-1))} \| a_{i+size-1..i} \| 0^{b\&(size-1)}$
    endfor
G.ROTL:
    for $i \leftarrow 0$ to 128-size by size
        $c_{i+size-1..i} \leftarrow a_{i+size-1-(b\&(size-1))..i} \| a_{i+size-1..i+size-1-(b\&(size-1))}$
    endfor
G.ROTR:
    for $i \leftarrow 0$ to 128-size by size
        $c_{i+size-1..i} \leftarrow a_{i+(b\&(size-1))-1..i} \| a_{i+size-1..i+(b\&(size-1))}$
    endfor
G.SHL:
    for $i \leftarrow 0$ to 128-size by size
        $c_{i+size-1..i} \leftarrow a_{i+size-1-(b\&(size-1))..i} \| 0^{b\&(size-1)}$
    endfor
G.SHR:
```

```
            for i ← 0 to 128-size by size
                $c_{i+size-1..i} \leftarrow a_{i+size-1}^{b\&(size-1)} \| a_{i+size-1..i+(b\&(size-1))}$
            endfor
        G.U.SHR:
            for i ← 0 to 128-size by size
                $c_{i+size-1..i} \leftarrow 0^{b\&(size-1)} \| a_{i+size-1..i+(b\&(size-1))}$
            endfor
    endcase
    case op of
        G.ADD, G.MUL, G.UMUL, G.DIV, G.UDIV:
        G.AND, G.OR, G.XOR, G.ANDN, G.NAND, G.NOR, G.XNOR, G.ORN,
        G.EXPAND G.U.EXPAND, G.SHL, G.SHR, G.U.SHR,
        G.GATHER, G.SCATTER, G.POLY:
            RegWrite(rc, 128, c)
        G.COMPRESS:
            RegWrite(rc, 64, c)
    endcase
enddef
```

Exceptions

Reserved Instruction

A 14